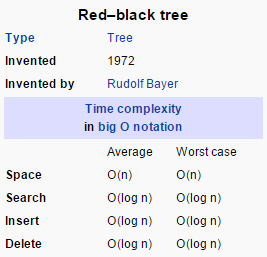
Symmetric Binary B-tree – Red Black tree

Rudolf Bayer (1972)

Self-balancing binary search tree

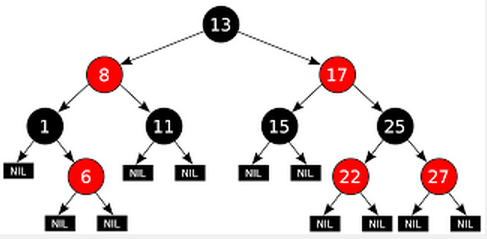
Each node is ‘painted’ either red or black such that certain properties are satisfied

The ‘certain properties’ limit the worst case unbalanced tree

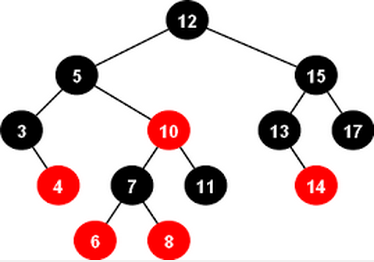


-Wikipedia

Storage need is 1 bit per node, which is often ‘free’



Beware the Wikipedia example above



This is also a red black tree

Symmetric Binary B-tree – Red Black tree

Leaves are null, not leaves like we have been thinking about them

This is predominantly about a worst-case guarantee

Rules:

All nodes are either red or black

The root node and all leaves are black

Every red node must have two black child nodes

Every path from a node to any of its leaves contain an equal number of black nodes

path length from root to furthest leaf <=

2 \* path length from root to nearest leaf

Symmetric Binary B-tree – Red Black tree

Why?

The shortest possible path has all black nodes, and the longest possible path alternates between red and black nodes

Functional programming languages for sets and associative arrays

Some Unix job schedulers

Computational geometry

Insert/delete are awkward, but tractable

Height Balanced AVL Tree

Velsky, Landis 1962

A self-balancing binary search tree

Self? Balancing is maintained during insert/delete

Let () == tree height, then

| (left child) – (right child) | <= 1



Height balanced, but not necessarily weight balanced